TUM

Analysis: Object Modeling

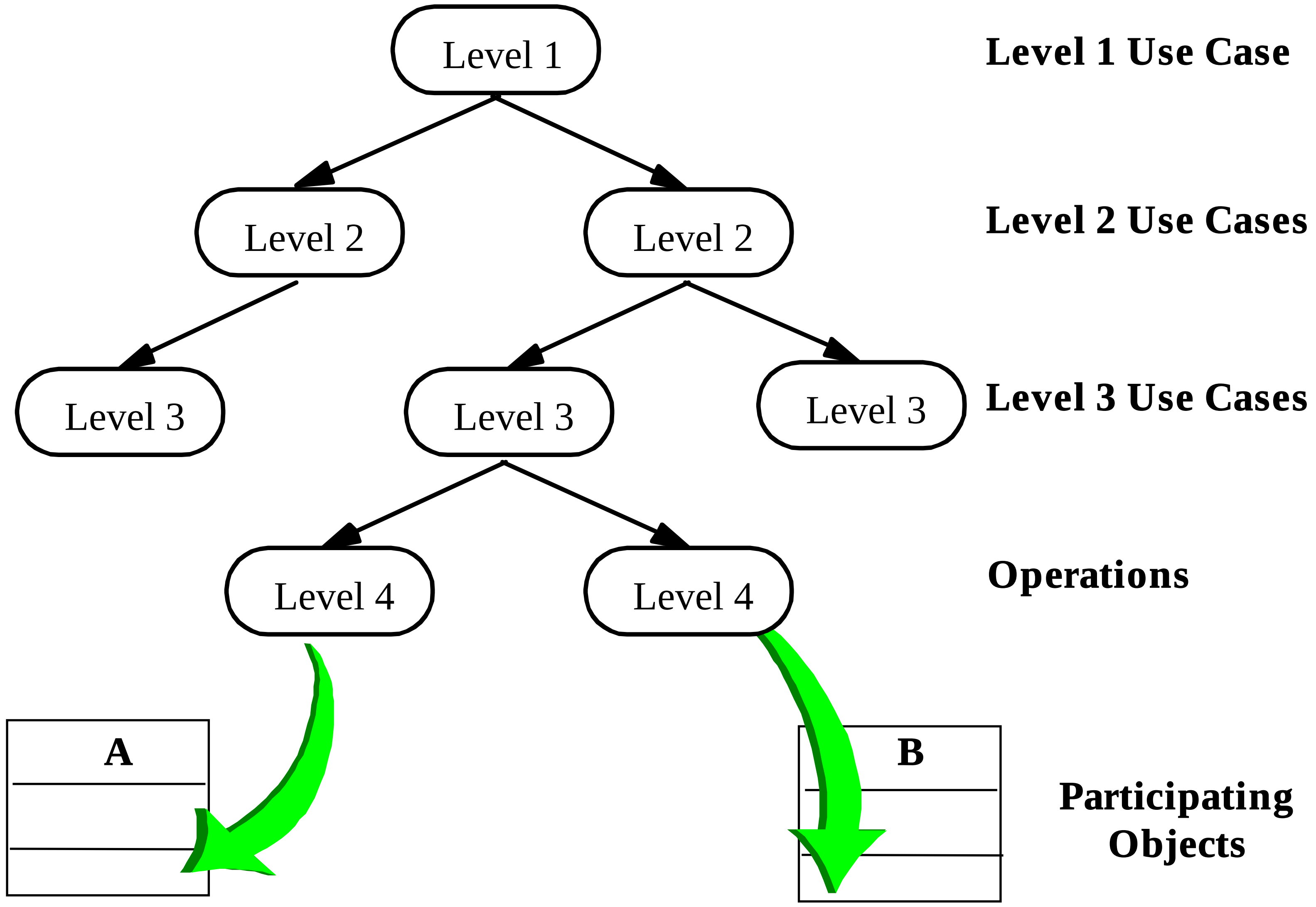
Bernd Brügge
Technische Universität München
Applied Software Engineering

15 November 2001

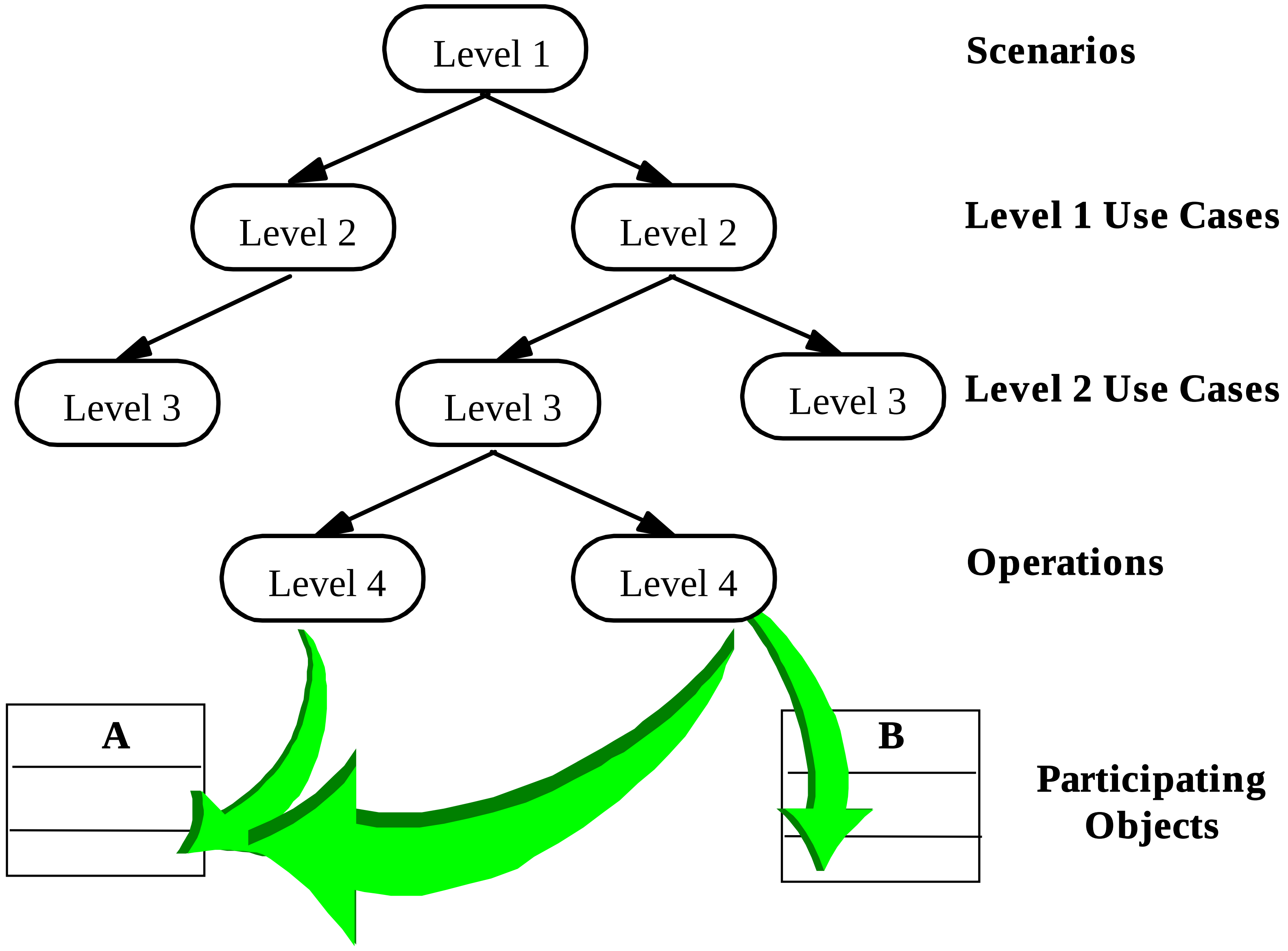
Outline

- **From use cases to class diagrams**
- **Model and reality**
- **A little discourse into philosophy**
- **Activities during object modeling**
- **Object identification**
- **Object types**
    - entity, boundary and control objects
- **Object naming**
- **Abott’s technique helps in object identification**
- **Users of class diagrams**

From Use Cases to Objects



From Use Cases to Objects: Why Functional Decomposition is not Enough



Reality and Model

- **Reality R:** Real Things, People, Processes happening during some time, Relationship between things
- **Model M:** Abstractions from (really existing or only thought of) things, people , processes and relationships between these abstractions.

Why models?

- **We use models**
  - To abstract away from details in the reality, so we can draw complicated conclusions in the reality with simple steps in the model
  - To get insights into the past or presence
  - To make predictions about the future

What is a "good" model?

- Relationships, which are valid in reality R, are also valid in model M.
  - **I : Mapping of real things in reality R to abstractions in the model M abbildet (Interpretation)**
  - **f_M: relationship between abstractions in M**
  - **f_R: relationship between real things inR**
- In a good model the following diagram is commutative:

$$\begin{tikzcd}
M \arrow[r, "f_M"] & M \\
R \arrow[u, "I"] \arrow[r, "f_R"'] & R \arrow[u, "I"']
\end{tikzcd}$$

Models are falsifiable

- **In the middle age people believed in truth**
- **Models of reality cannot be true**
- **A model is always an approximation**
  - We must say “according to our knowledge”, or “with today’s knowledge”
- **Popper (“Objective Knowledge):**
  - We can only build models from reality, which are “true” until, we have found a counter example ***(Principle of Falsification)***
    - And even then we might stick with the model (“because it works quite well in most settings”)
- **The falsification principle is the basis of software development**
  - The goal of prototypes, reviews and system testing is to falsify the software system

Models of models of models...

- **Modeling is relative.** We can think of a model as reality and can build another model from it (with additional abstractions).

$$\begin{tikzcd}
 & \cdots & \\
M_2 \arrow[rr, "f_{M2}"] & & M_2 \\
M_1 \arrow[u, "\text{Analysis}"] \arrow[rr, "f_{M1}"] & & M_1 \arrow[u, "I_2"'] \\
R \arrow[u, "\text{Requirements Elicitation}"] \arrow[rr, "f_R"'] & & R \arrow[u, "I_1"']
\end{tikzcd}$$

A small discourse into Philosophy

- **Philosophy works on 3 major problems**
  - ***Metaphysics:*** What is reality?
  - ***Epistemology:*** What is knowledge? How can we store knowledge in our brain? How far can I describe reality with knowledge?
  - ***Ethics:*** What is good, what is bad?
- **Metaphysics and epistemology depend on each other:**
  - Assertions about reality depend on closely on assertions about knowledge and vice versa.
- **Relationship to software engineering**
  - Metaphysics <=> Modeling
  - Epistemology <=> Acquisition of knowledge, knowledge management
  - Ethics: <=> Good and bad practices during software development

The four basic questions in metaphysics

1. **Is reality real or not real?**

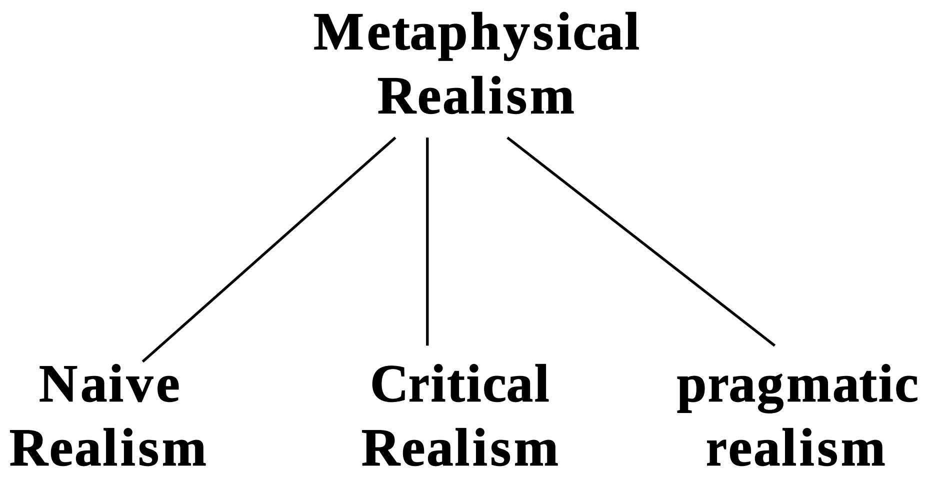
    Does reality exist only in our brain or does it exist independently from our existence?

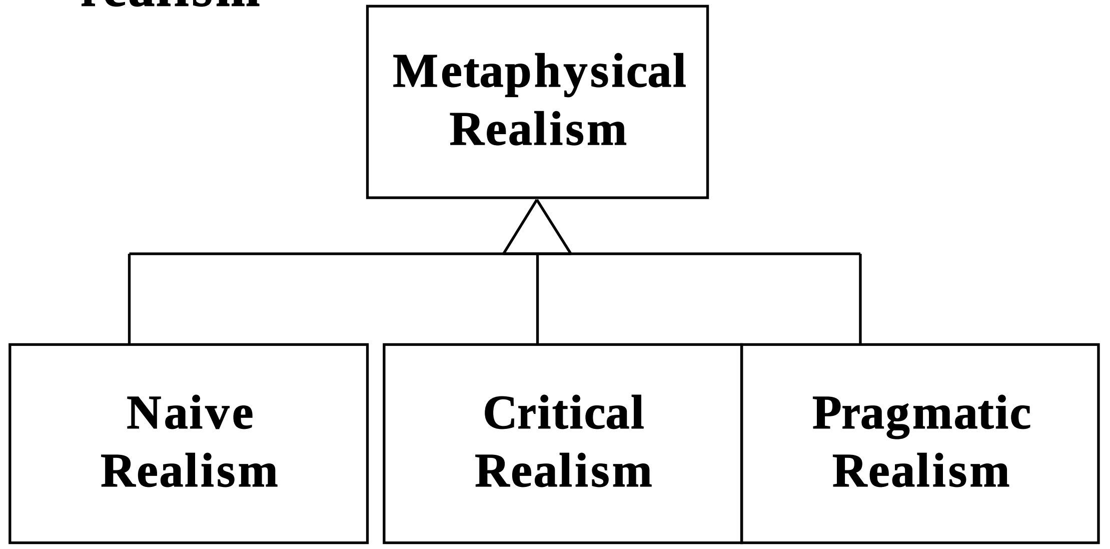
2. **What is reality made out of?**
3. **How many realities are there (1,2, many)?**
4. **Is reality constant or does it change?**

1. Reality: Real or ideal?

- **The metaphysical realism assumes, that reality is real**
  - Reality exists outside our brain. It is “really” real. Subtypes of Realism:
    - **Naïve realism:** Things are real, that is a fact!
    - **Critical realism** (transzendental Realism): Things are real, but I see only what I want to see
    - **Pragmatisc realism:** Realism works, that’s why reality is real
- **The metaphysical idealism assumes that reality is an illusion.**

Categorization of the various types of realism

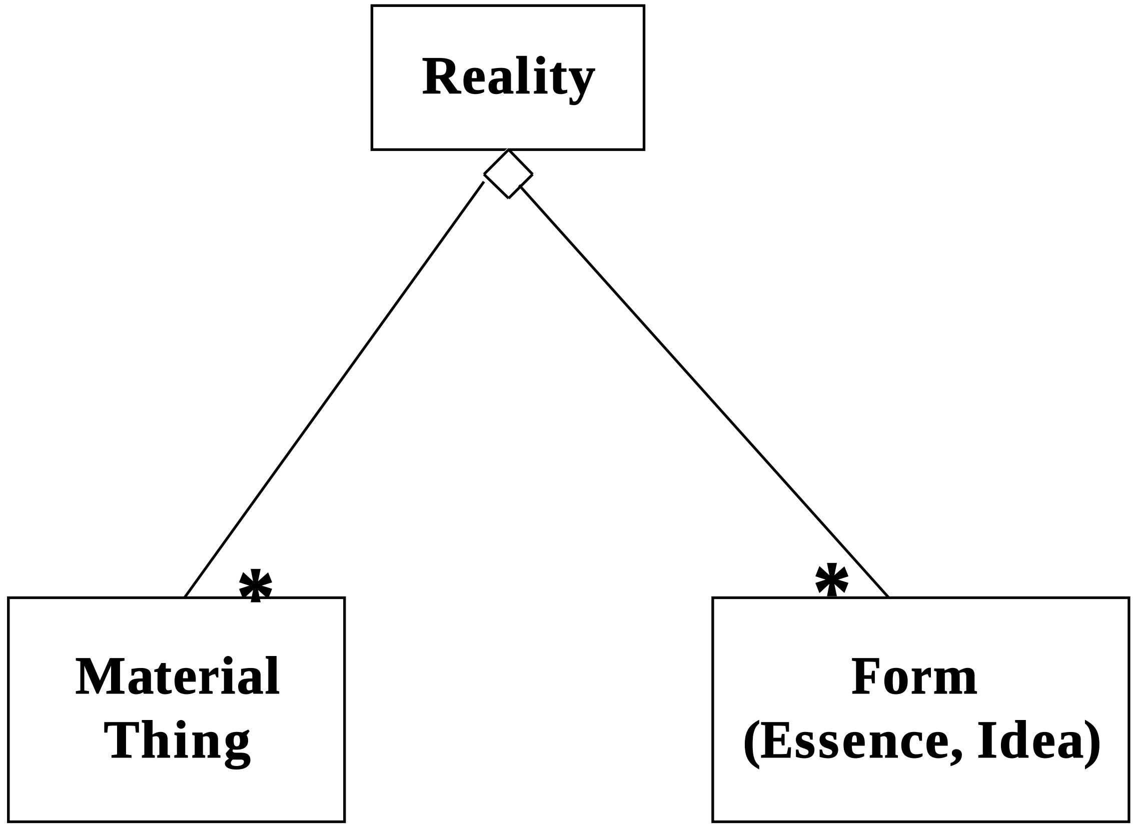




2. What is reality made out of?

- **Materialism:**
  - Reality consists of real things
  - ***Sokrates*****:** Everything is made out of water
- **Antimaterialism:**
  - Reality consists of real things as well as of ideas
  - ***Plato***: A form,e.g beauty, is as real as real things, e.g. This little train(actually forms are more real, because they are permanent, real things live only for a short time)
- **Scientific materialism**:
  - Reality consists only of things that have energy and/or mass
  - ***Modern science*****:** mind-reading capability is not real

Model of Plato’s Antimaterialism



3. How many realities are there ?

- **Monism:**
  - There is only one thing, which is simultaneously the source and essence of reality (***Thales von Milet***: Everything is made out of water)
- **Dualism:**
  - There are 2 different sources for things in Reality
  - ***Plato***: Forms and Material Things are 2 types of Reality
  - ***Descartes:*** The mind and the body are separate things
  - ***Tao:*** Each thing consists of two complementary principles: Ying und Yang
- **Pluralism:**
  - ***Software Engineering:*** There are many realities , the customer requirements are reality

4. Is reality constant or does it change?

- **Parmenides (600 A.D):**
  - There is a difference between appearance and underlying reality. Change is an illusion, reality is constant
- **Heraklit (540-475 A.D.):**
  - Everything flows, there is no solid substance
    - “Jupiter’s eye” is actually a hurricane
    - Modern physics: Reality is a field of vibrations
- **Software Engineering:**
  - The graphical user interface (“GUI”) changes, but the underlying business process is constant.
  - WIMP : Windows, Icons, Mouse and Pointing Device
  - The business process changes as result of technology enablers: “Change is the only constant” (Hammer&Champy, Reengineering)

The 4 basic questions in epistemology

- 1. How do we acquire knowledge, through our senses or through our intelligence?
- 2. How far can we describe or create reality with knowledge?
- 3. What is knowledge made out of?
- 4. What are the activities during knowledge acquisition?

1. How do we acquire knowledge?

- **Empirism:** Knowledge is acquired by experimentation and through our senses
  - **Our brain is initially empty (“tabula rasa”)**
- **Rationalism:** Knowledge is acquired by our mind
  - **The brain is already at birth equipped with ideas (“a priori”)**
- **Voluntarism:** Knowledge is only acquired if you want to achieve something
- **Intuitionism:** Knowledge is acquired by intuition

Taxonomy of knowledge acquistion methods

Knowledge Acquisition

Realism:

• Koncepts - fact as well as a priori conzepts- are not simply copies or extensions of the sensual experience

• Concepts are built into our mind:

- •Concepts are “remembrance” of forms. They can be triggered by senses, but they are already in our mind, they are only woken up. (Plato)
- •Concepts are categories of our mind. They are structures which allow us mentally to keep track of sensual objects. Concepts are not derived from sensor data, but are used to make sense from sensor data (Kant)

d empirically.
but such concepts
lity. Example: It is
triangle add up to to
that that there are
that we can find

Can we describe reality with knowledge?

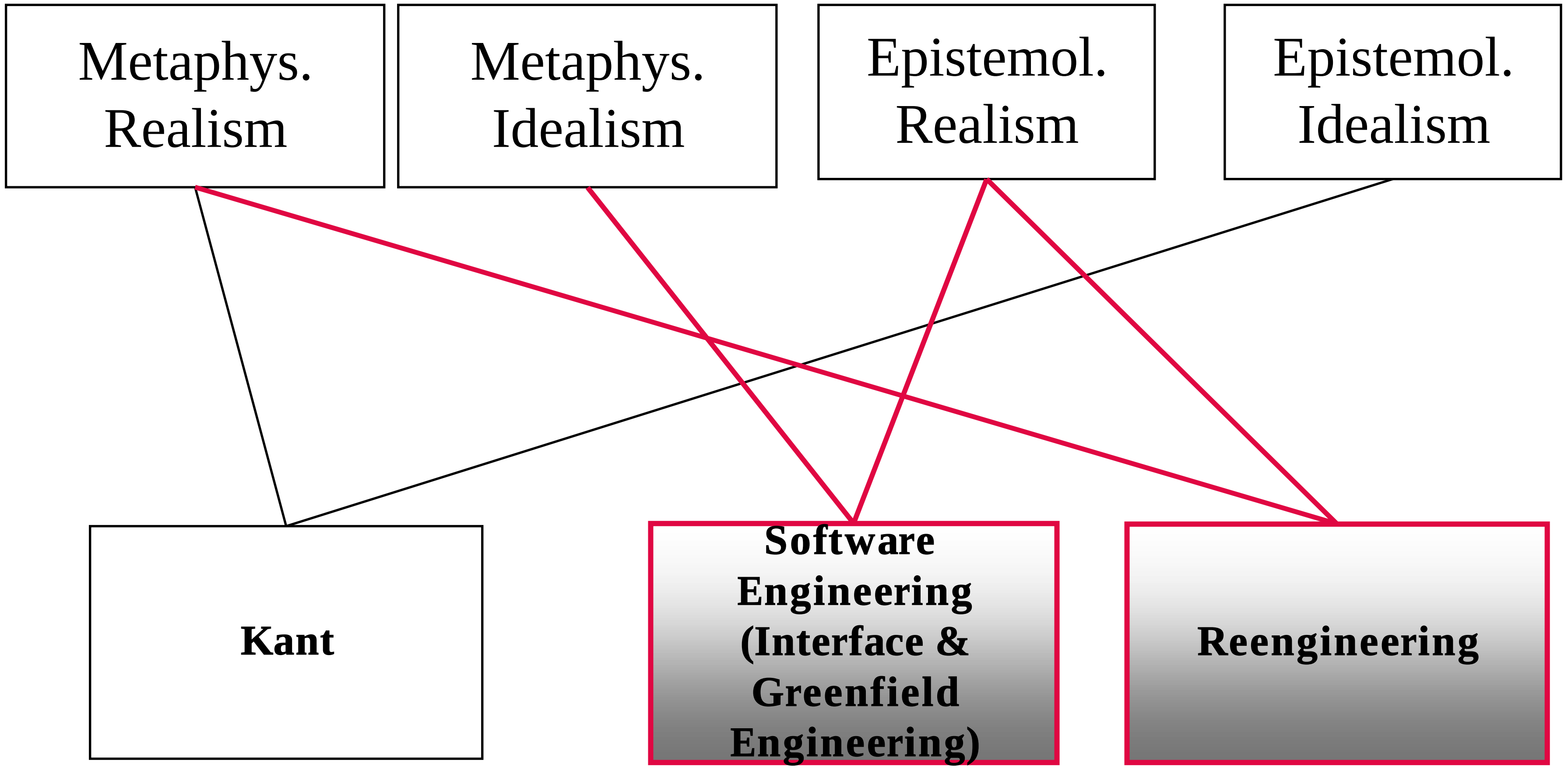
- **Epistemological idealism:**
  - What you know about an object, exists only in your mind. Models can only describe parts of reality, never reality.
- **Epistemological realism:**
  - The knowledge about an object is independent from our mind. Models can describe reality.

- ***Epistemological idealists are pessimists:***
  - There are always conclusions, that you cannot draw in the model, because they depend on components in reality which are not described in the model.
- ***Epistemological realists are optimists:***
  - All conclusions in the model describe things in reality.

Combining metaphysics and ephistemology

- **Metaphysical realist, ephistemological realist:**
  - There is a reality outside of my mind, I can acquire knowledge about this reality and I can represent reality with my model. (Software Engineering: Reengineering)
- **Metaphysiscal realist, ephistemological idealist:**
  - There is a reality outside of my mind, the knowledge about this reality is limited by the structures and activities of my mind (Kant)
- **Metaphysisical idealist, ephistemological idealist:**
  - Reality depends on a (another) mind, my knowledge about this reality is limited by my mind.
- **Metaphysiscal idealist, ephistemological realist:**
  - Reality depends on a (another) mind, my mind can understand the concepts of this other mind, and I can represent this externally with models (Software Engineering: Customer specifies the system)

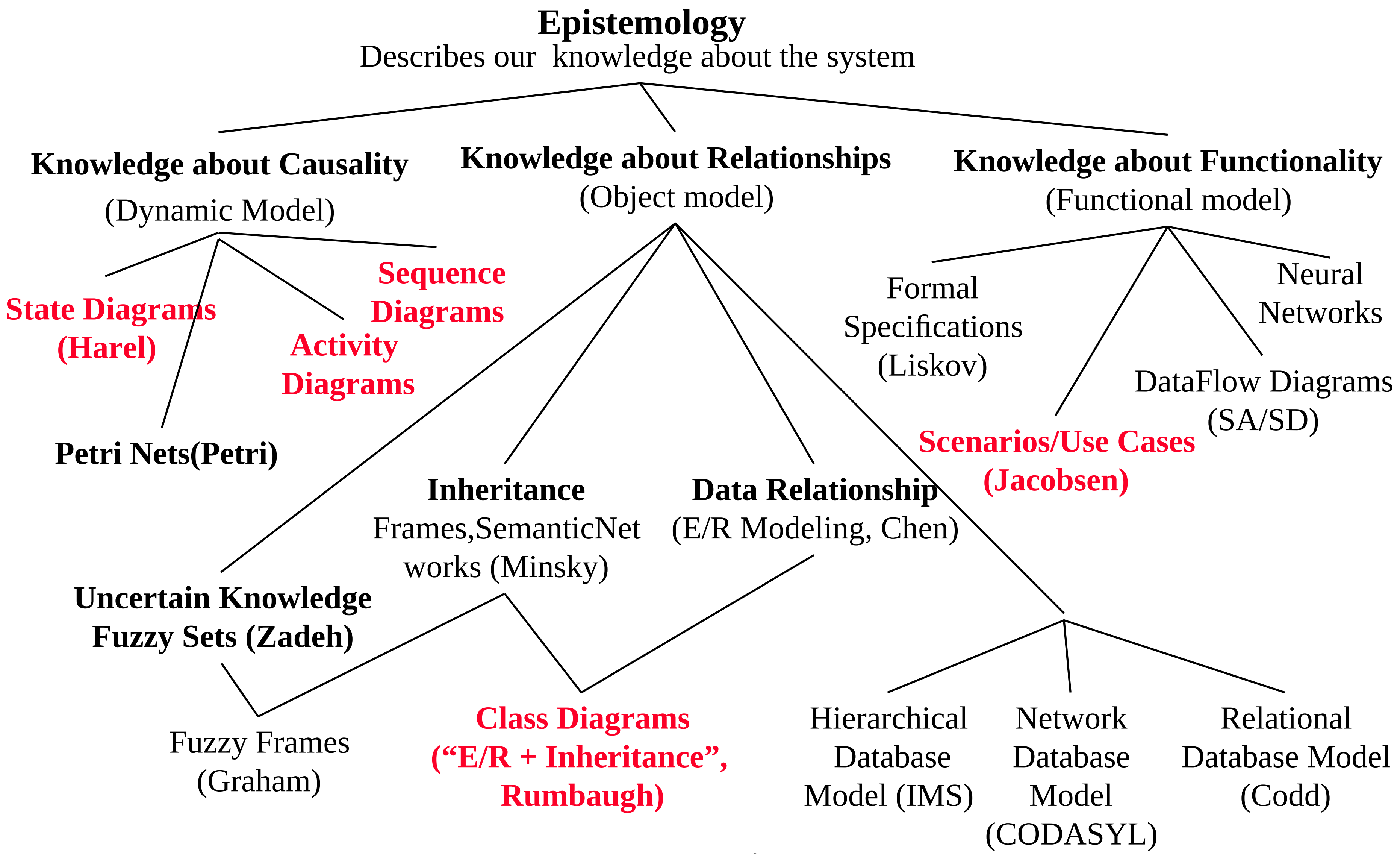
Combination of metaphysics and ephistemology



Realities for software engineers

- Some people say: “The computer scientist can play god, because they can create realities”. Nonsense.
- But : The computer scientist can model different kinds of realities and build them:
  - **An existing system (physical system, technical system, social system, software system)**
    - **An important special case is here when the existing system is a software system. We then call it “Legacy System”**
  - **An idea without counterpart in reality:**
    - **A visionary scenario or a customer requirement.**
- The constructed reality might actually only be part of the ideas, namely those that were realizable in software
  - **Example: A visionary scenario turns out to be a dream, a customer requirement turns out to be too expensive to realize.**

How do we model complex systems (Natural Systems, Social Systems, Artificial Systems)?


Epistemology
Describes our knowledge about the system
Knowledge about Causality
(Dynamic Model)
Knowledge about Relationships
(Object model)
Knowledge about Functionality
(Functional model)
State Diagrams (Harel)
Sequence Diagrams
Activity Diagrams
Petri Nets(Petri)
Formal Specifications (Liskov)
Neural Networks
DataFlow Diagrams (SA/SD)
Scenarios/Use Cases (Jacobsen)
Inheritance
Frames,SemanticNet works (Minsky)
Data Relationship
(E/R Modeling, Chen)
Uncertain Knowledge Fuzzy Sets (Zadeh)
Fuzzy Frames (Graham)
Class Diagrams ("E/R + Inheritance", Rumbaugh)
Hierarchical Database Model (IMS)
Network Database Model (CODASYL)
Relational Database Model (Codd)

Activities during Object Modeling

- **Main goal: Find the important abstractions**
- **What happens if we find the wrong abstractions?**
  - Iterate and correct the model
- **Steps during object modeling**
  - 1. Class identification
    - Based on the fundamental assumption that we can find abstractions
  - 2. Find the attributes
  - 3. Find the methods
  - 4. Find the associations between classes
- **Order of steps**
  - Goal: get the desired abstractions
  - Order of steps secondary, only a heuristic
  - Iteration is important

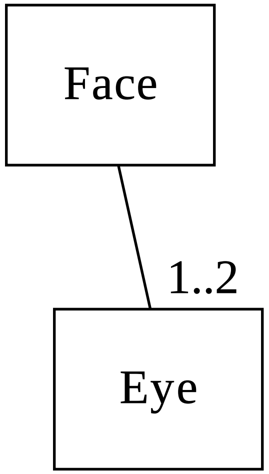
Class Identification

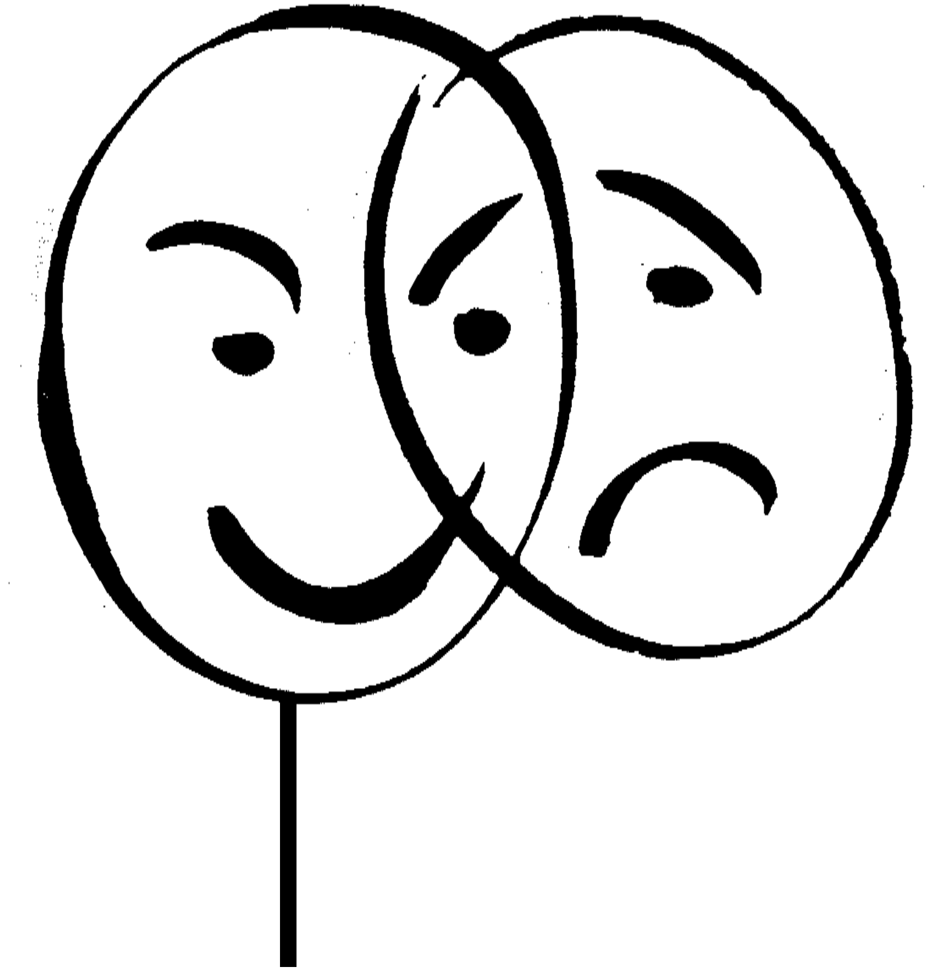
- **Identify the boundaries of the system**
- **Identify the important entities in the system**
- **Class identification is crucial to object-oriented modeling**
- **Basic assumption:**
  - 1. We can find the classes for a new software system (Forward Engineering)
  - 2. We can identify the classes in an existing system (Reverse Engineering)
- **Why can we do this?**
  - Philosophy, science, experimental evidence

Class identification is an ancient problem

- **Objects are not just found by taking a picture of a scene or domain**
- **The application domain has to be analyzed.**
- **Depending on the purpose of the system different objects might be found**
  - How can we identify the purpose of a system?
  - Scenarios and use cases
- **Another important problem: Define system boundary.**
  - What object is inside, what object is outside?

What is This?





Modeling in Action

- **Face**
- **Mask**
- **Sad**
- **Happy**
- **Is it one Face or two?**
- **Who is using it?**
  - Person at Carneval?
  - Bankrobber?
  - Painting collector
- **How is it used?**

Pieces of an Object Model

- **Classes**
- **Associations (Relations)**
  - Generic associations
  - Canonical associations
    - Part of- Hierarchy (Aggregation)
    - Kind of-Hierarchy (Generalization)
- **Attributes**
  - Detection of attributes
  - Application specific
  - Attributes in one system can be classes in another system
  - Turning attributes to classes
- **Operations**
  - Detection of operations
  - Generic operations: Get/Set, General world knowledge, design patterns
  - Domain operations: Dynamic model, Functional model

Object vs Class

- **Object (instance): Exactly one thing**
  - This lecture on Software Engineering on November 15 from 14:30 - 16:00
- **A class describes a group of objects with similar properties**
  - Game, Tournament, mechanic, car, database
- ***Object diagram*****: A graphic notation for modeling objects, classes and their relationships ("associations"):**
  - *Class diagram:* **Template for describing many instances of data. Useful for taxonomies, patters, schemata...**
  - *Instance diagram:* **A particular set of objects relating to each other. Useful for discussing scenarios, test cases and examples**
- **Together-J: CASE (Computer-Aided Software Engineering) Tool for building object diagrams, in particular class diagrams**
  - Lecture tomorrow (November 16)

Class identification

- **Finding objects is the central piece in object modeling**
- **Approaches**
  - Application domain approach (not a special lecture, examples):
    - Ask application domain expert to identify relevant abstractions
  - Syntactic approach (today):
    - Start with use cases. Extract participating objects from flow of events
    - Use noun-verb analysis (Abbot’s technique) to identify components of the object model
  - Design patterns approach (Lecture on design patterns)
    - Use reusable design patterns
  - Component-based approach (Lecture on object design):
    - Identify existing solution classes

How do you find classes?

- **Finding objects is the central piece in object modeling**
  - Learn about problem domain: Observe your client
  - Apply general world knowledge and intuition
  - Take the flow of events and find participating objects in use cases
  - Try to establish a taxonomy
  - Do a syntactic analysis of *problem statement*, *scenario* or *flow of events*
  - Abbott Textual Analysis, 1983, also called noun-verb analysis
    - Nouns are good candidates for classes
    - Verbs are good candidates for opeations
  - Apply design knowledge:
    - Distinguish different types of objects
    - Apply design patterns (Lecture on design patterns)

How do you find classes?

- **Finding objects is the central piece in object modeling**
    - Learn about problem domain: Observe your client
    - Apply general world knowledge and intuition
    - Take the flow of events and find participating objects in use cases
    - Try to establish a taxonomy
    - Apply design knowledge:
        - Distinguish different types of objects
        - Apply design patterns (Lecture on design patterns)
    - Do a syntactic analysis of *problem statement*, *scenario* or *flow of events*
    - Abbott Textual Analysis, 1983, also called noun-verb analysis
        - Nouns are good candidates for classes
        - Verbs are good candidates for opeations

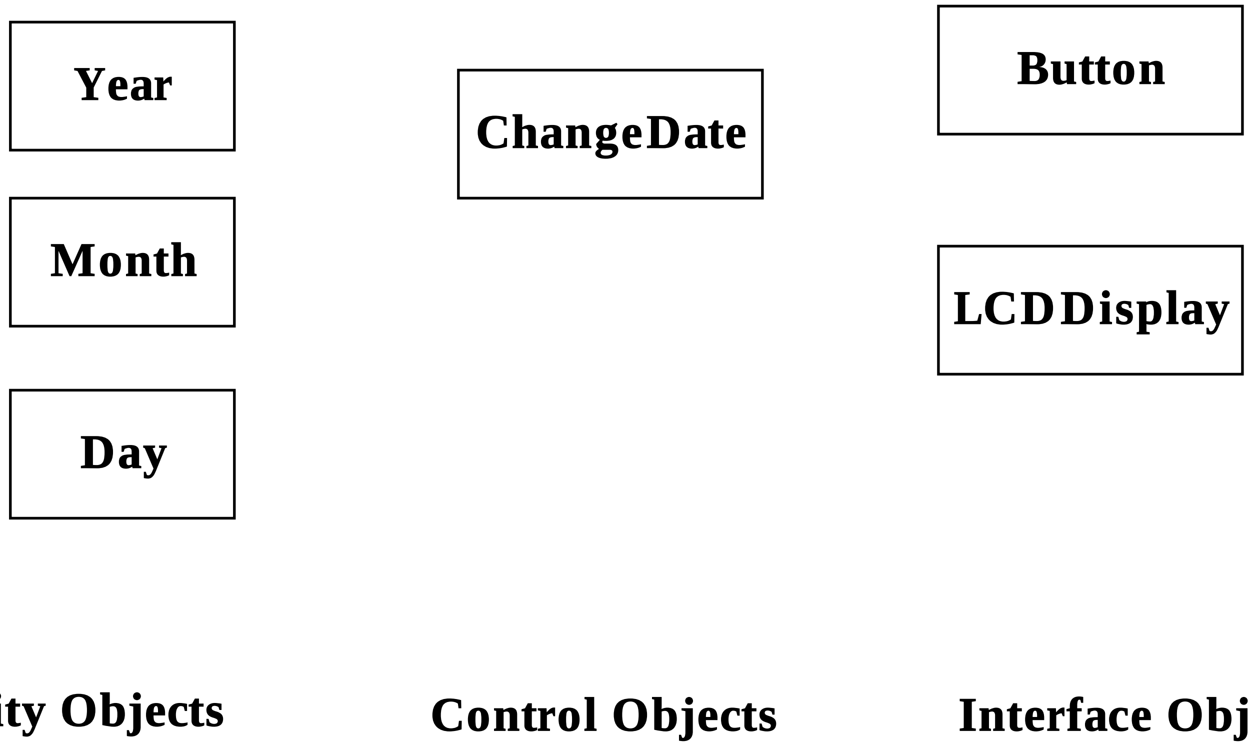
Finding Participating Objects in Use Cases

- **Pick a** *use case* **and look at its** *flow of events*
  - Find terms that developers or users need to clarify in order to understand the flow of events
  - Look for recurring nouns (e.g., Incident),
  - Identify real world entities that the system needs to keep track of (e.g., FieldOfficer, Dispatcher, Resource),
  - Identify real world procedures that the system needs to keep track of (e.g., EmergencyOperationsPlan),
  - Identify data sources or sinks (e.g., Printer)
  - Identify interface artifacts (e.g., PoliceStation)
- **Be prepared that some objects are still missing and need to be found:**
  - Model the flow of events with a sequence diagram
- **Always use the user's terms**

Object Types

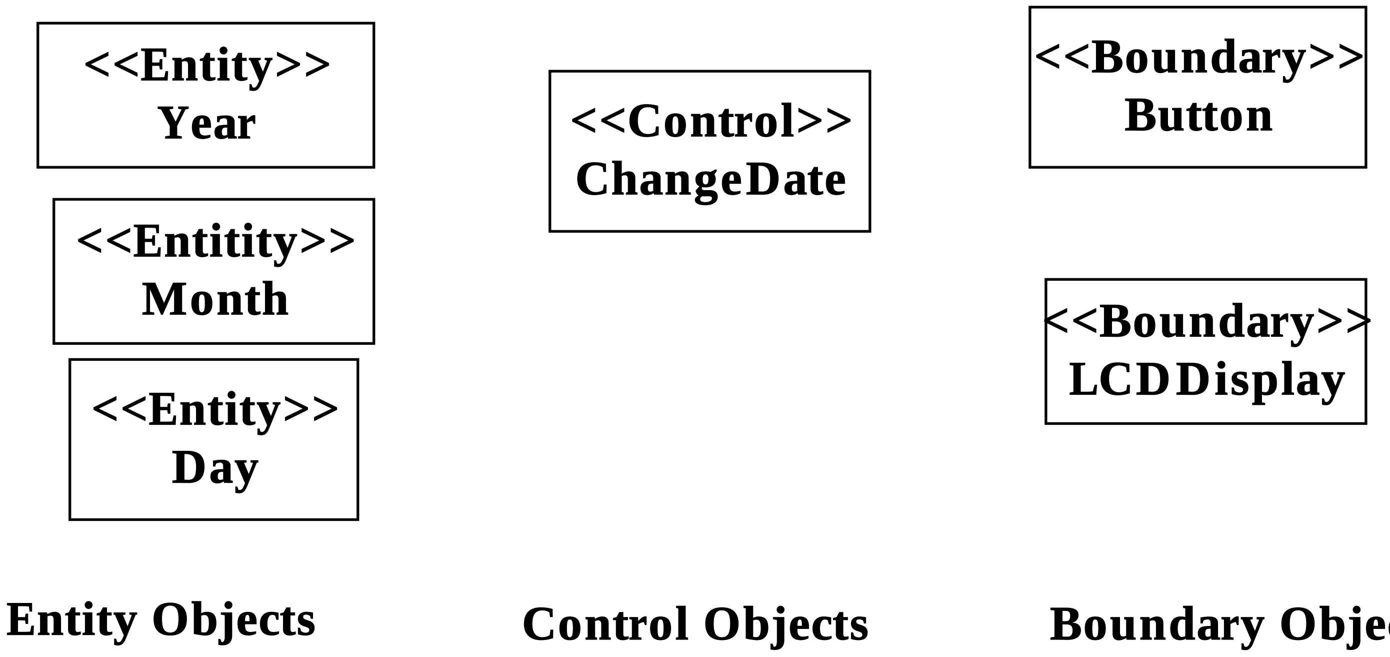
- **Entity Objects**
  - Represent the persistent information tracked by the system (Application domain objects, “Business objects”)
- **Boundary Objects**
  - Represent the interaction between the user and the system
- **Control Objects:**
  - Represent the control tasks performed by the system
- **Having three types of objects leads to models that are more resilient to change.**
  - The interface of a system changes more likely than the control
  - The control of the system change more likely than the application domain
- **Object types originated in Smalltalk:**
  - Model, View, Controller (MVC)

Example: 2BWatch Objects



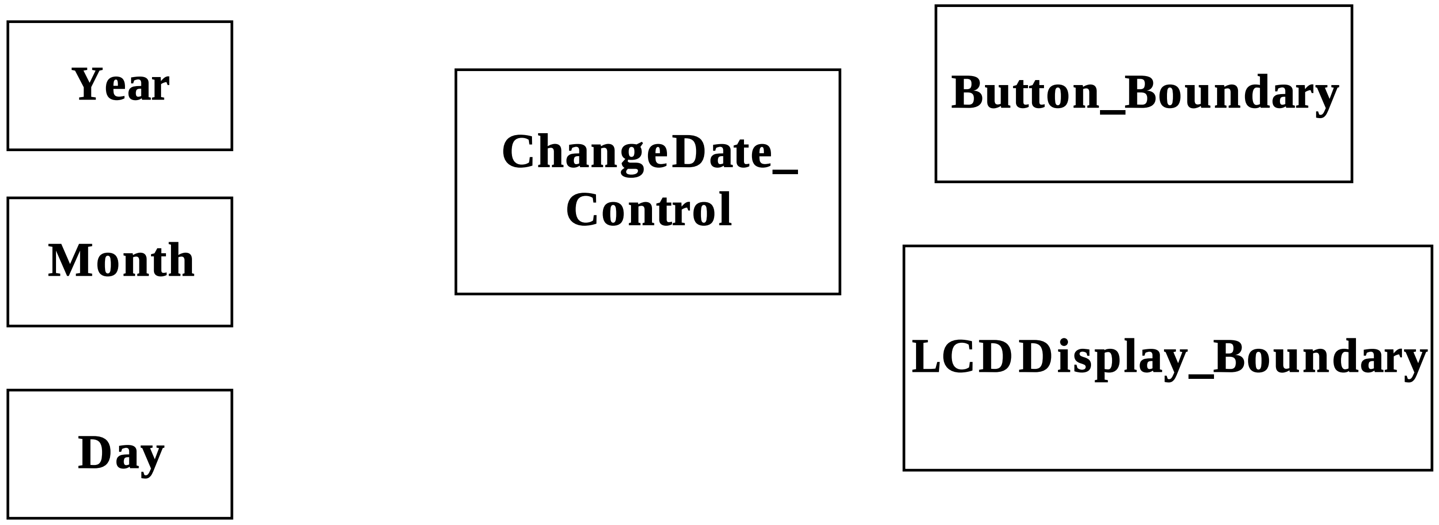
Naming of Object Types in UML

- UML provides several mechanisms to extend the language
- UML provides the stereotype mechanism to present new modeling elements



Recommended Naming Convention for Object Types

- **To distinguish the different object tpyes on a syntactical basis, we recommend suffixes:**
- **Objects ending with the "_Boundary" suffix are boundary objects**
- **Objects ending with the "_Control" suffix are control objects**
- **Entity objects do not have any suffix appended to their name.**



Example: Flow of events

- **The customer enters a store with the intention of buying a toy for his child with the age of n.**
- **Help must be available within less than one minute.**
- **The store owner gives advice to the customer. The advice depends on the age range of the child and the attributes of the toy.**
- **The customer selects a dangerous toy which is kind of unsuitable for the child.**
- **The store owner recommends a more yellow doll.**

Mapping parts of speech to object model components [Abbot 1983]

Part of speech	*Model component*	*Example*
Proper noun	object	Jim Smith
Improper noun	class	Toy, doll
Doing verb	method	Buy, recommend
being verb	inheritance	is-a (kind-of)
having verb	aggregation	has an
modal verb	constraint	must be
adjective	attribute	3 years old
transitive verb	method	enter
intransitive verb	method (event)	depends on

Another Example

Flow of events:

- The customer enters the store to buy a toy.
- It has to be a toy that his daughter likes and it must cost less than 50 Euro.
- He tries a videogame, which uses a data glove and a head-mounted display. He likes it.
- An assistant helps him.
- The suitability of the game depends on the age of the child.
- His daughter is only 3 years old.
- The assistant recommends another type of toy, namely the boardgame "Monopoly".

Is this a good use Case?

Not quite!

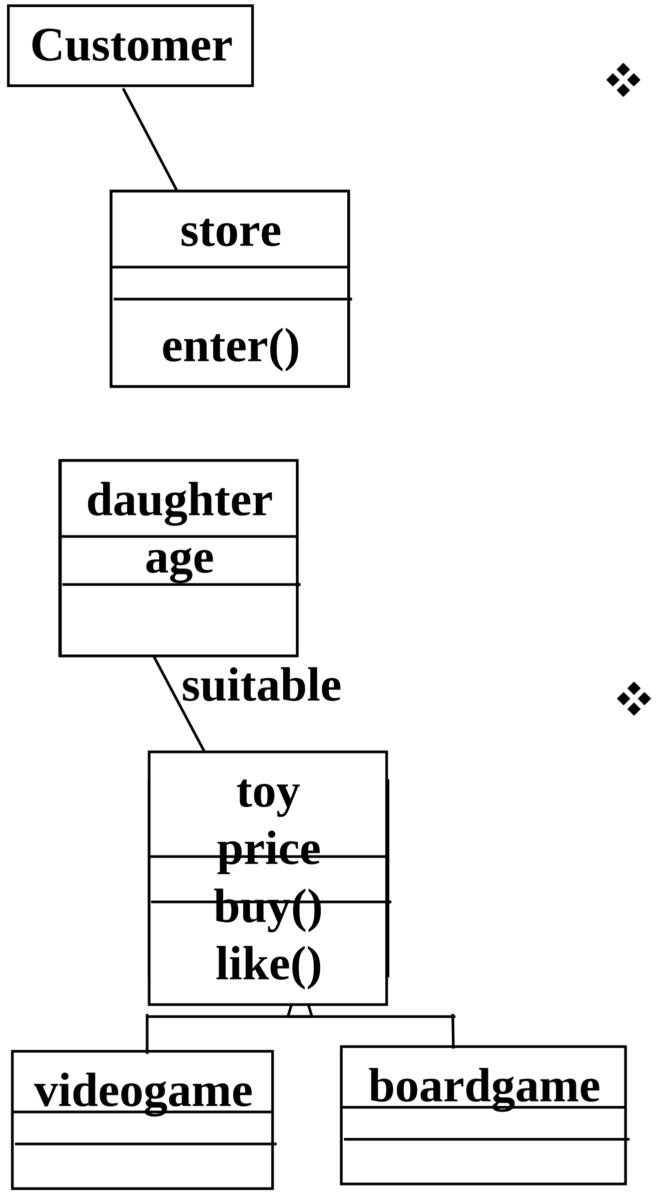
"Monopoly" is probably a left over from the scenario

The use case should terminate with the customer leaving the store

Textual Analysis using Abbot‘s technique

Example	*Grammatical construct*	*UML Component*
“Monopoly"	Concrete Person, Thing	Object
“toy"	noun	class
"3 years old"	Adjective	Attribute
“enters"	verb	Operation
“depends on…."	Intransitive verb	Operation (Event)
“is a" ,“either..or", “kind of…"	Classifying verb	Inheritance
"Has a ", “consists of"	Possessive Verb	Aggregation
“must be", “less than…"	modal Verb	Constraint

Generation of a class diagram from flow of events



Flow of events:

- The customer enters the store to buy a toy. It has to be a toy that his daughter likes and it must cost less than 50 Euro. He tries a videogame, which uses a data glove and a head-mounted display. He likes it.

- An assistant helps him. The suitability of the game depends on the age of the child. His daughter is only 3 years old. The assistant recommends another type of toy, namely a boardgame. The customer buy the game and leaves the store

Order of activities in modeling

1. **Formulate a few scenarios with help from the end user and/or application domain expert.**
2. **Extract the use cases from the scenarios, with the help of application domain expert.**
3. **Analyse the flow of events, for example with Abbot's textual analysis.**
4. **Generate the class diagrams, which includes the following steps:**
   1. Class identification (textual analysis, domain experts).
   2. Identification of attributes and operations (sometimes before the classes are found!)
   3. Identification of a*ssociations between classes*
   4. Identification of multiplicities
   5. Identification of roles
   6. Identification of constraints

Ways to find objects

- Syntactical investigation with Abbot‘s techniqe**:**
  - In the problem statement (originally proposed, but rarely works if the problem statement is large (more than 5 pages)
  - In the flow of events of use cases
  - => Textuelle Analyse nach Abbot
- Use of various knowledge sources:
  - Application knowledge: Interviews of end users and experts, to determine the abstractions of the application domain.
  - *Design knowledge:* Reusable abstractions in the solution domain.
  - General world knowledge: Also use your generic knowledge and intution.
- Formulation of scenarios **(in natural language):**
  - Description of the concrete usage of the system.
- Formulation of use cases **(natural language and UML):**
  - Description of functions with actors and flow of events

Summary

- **Modeling vs reality**
- **System modeling**
  - Object model
  - Dynamic model
  - Functional model
- **Object modeling is the central activity**
  - Class identification is a major activity of object modeling
- **Abbot’s technique**
  - syntactic rules to find classes / objects